

Toward fully autonomous mobile manipulation for industrial environments

Andreas Dömel¹, Simon Kriegel¹, Michael Kaßecker¹,
Manuel Brucker¹, Tim Bodenmüller¹ and Michael Suppa²

Abstract

This work presents a concept for autonomous mobile manipulation in industrial environments. Utilizing autonomy enables an unskilled human worker to easily configure a complex robotics system in a setup phase before carrying out fetch and carry operations in the execution phase. In order to perform the given tasks in real industrial production sites, we propose a robotic system consisting of a mobile platform, a torque-controlled manipulator, and an additional sensor head. Multiple sensors are attached which allow for perception of the environment and the objects to be manipulated. This is essential for coping with uncertainties in real-world application. In order to provide an easy-to-use and flexible system, we present a modular software concept which is handled and organized by a hierarchical flow control depending on the given task and environmental requirements. The presented concept for autonomous mobile manipulation is implemented exemplary for industrial manipulation tasks and proven by real-world application in a water pump production site. Furthermore, the concept has also been applied to other robotic systems and other domains for planetary exploration with a rover.

Keywords

Manufacturing, mobile manipulation, autonomous systems, perception

Date received: 25 November 2016; accepted: 25 May 2017

Topic: Special Issue - Robot Perception for Manufacturing

Topic Editor: Stefano Ghidoni

Associate Editor: Emanuele Menegatti

Introduction

The remarkable progresses in developing and interconnecting sensors, machines and people in the past years led to the idea of fusing high-tech networking know-how with state-of-the-art robot and automation technology into a smart manufacturing solution. This kind of advanced interoperability—an Internet of Things—is not just about connecting smart devices with the obvious goal of improving manufacturing execution systems or enterprise resource planning. It enables higher levels of automated interaction such as approaching product development from a new perspective and permitting the customer to self tailor products.

Currently, it is a frequent practice to custom design part feeding, robot programming, and process setup for each production step. This requires a major effort and is only possible by employing human experts to develop a

specialized solution for each problem. Large quantity production allows for a considerable large amount of money to invest. With respect to marginal return in limited lots, setting up an expensive single production line may not be profitable.

Along with altered surrounding conditions, the system design priorities change. The critical factors shift from the robot's execution speed to the system's flexibility and

¹Robotics and Mechatronics Center, German Aerospace Center (DLR), Wessling, Germany

²Roboception GmbH, München, Germany

Corresponding author:

Andreas Dömel, Robotics and Mechatronics Center, German Aerospace Center (DLR), Münchener Str. 20, 82234 Wessling, Germany.

Email: andreas.doemel@dlr.de



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License

(<http://www.creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

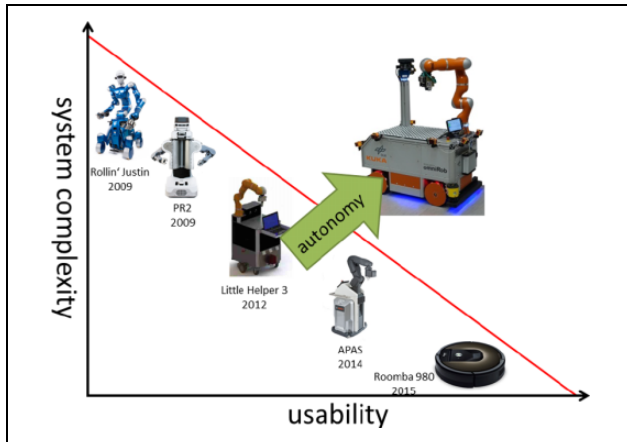


Figure 1. Overcoming the contradiction of system complexity and simple usability requires autonomy.

usability due to the following reasons. First, the robot's environment is not static and there are no fences and no part suppliers which guarantee for a clearly defined setup. Hence, the robot's surroundings are considered not to be extremely dynamic—they tend to change slightly each time the robot approaches a station. Second, having a mobile robot is crucial in order to work at different stations and tasks. Clearly, coping with this problem requires sophisticated sensor data processing and perception capabilities as vital component of the system. This in turn raises the complexity of the system to a much higher level in comparison to a traditional robotic system in industrial applications. As flexibility may be the unique selling point of a small business, new tasks and changes in the production are part of the usual work flow. In vastly automated facilities, those adaptations are done by system integrators and engineers providing the expertise to handle a complex system. In a small company, an employed robotic expert may not be cost-efficient, so any adept blue-collar worker should be able to set up the system. As depicted in Figure 1, there are highly complex systems such as the PR2¹ or DLR Justin² which could solve complex tasks, but programming those systems is challenging even for robotic experts. On the other hand, there are systems which can be used by an unskilled worker such as the iRobot Roomba or the Bosch APAS but those systems are not able to solve, for example, fetch and carry tasks.

In our work, ease-of-use of a complex robotic system is achieved by increasing the system's autonomy. However, the autonomy needed in the industrial domain differs from autonomy, for example, in service robotics. In production, automated or manual, two clearly separated phases (see Figure 2) can be identified.

In the *setup phase*, workers are trained for their job or robots are programmed for their tasks. In the industrial domain, tasks are well documented in detail. Thus, human workers have to follow detailed working descriptions and they are trained to do each step exactly the way it is

documented. The second phase is the *execution phase* in which goods are produced by workers or robots. Usually the tasks are executed repetitively.

The goal of our approach is to reduce the effort in setup phase of the robot to the level of a human worker's training for a comparable task and to raise the robustness of task execution to a comparable level of a human worker.

In case of a very user-friendly system, any unskilled worker is able to program the robot for a new task. This worker knows the task but has no expertise in robotics. Therefore, we propose using this information during task training instead of utilizing complex reasoning approaches. In this case, the worker has to be supported by the system, solving the robotic part of the problems itself, which requires a superior level of autonomy. Then, during execution phase, the robotic system is operating fully autonomously. The task is fixed but the system has to cope with uncertainties and changes in the environment. Therefore, perception and path planning are vital components of our system.

We present a concept toward fully autonomous mobile manipulation focusing on fetch and carry operations as important representative of industrial tasks. Keypoints of our approach are:

- modularization: break down of the system's functionality into small functional units;
- hierarchical flow control: coordination of modules for implementing complex behaviors;
- perception: sensing the environment to cope with uncertainties and changes;
- knowledge representation: abstract representation of the state of the task, the environment, and the system; and
- two phase approach: application of the given characteristics of tasks in the industrial domain.

To describe the concept for autonomous mobile manipulators, we refer exemplarily to the DLR omniRob system (see Figure 3, top). Nevertheless, the concept has been applied to different platforms such as in Figure 3 (bottom).

This article is organized as follows. In the next section, we give an overview of the related work. Then, we describe the hardware and software concept of the robotic system. In the following section, modules which enable the robot to generate knowledge about its environment and tasks during the setup phase are introduced. These modules are partially very time-consuming but they are not needed during task execution. Most of the setup phase is covered by these knowledge-generating modules. During execution, modules for perception and cognition are utilized and presented. Keeping track of the current world state and reacting to changes requires efficient algorithms to keep the execution time acceptable. Then, the application of the methods described in the previous chapters is presented. The system

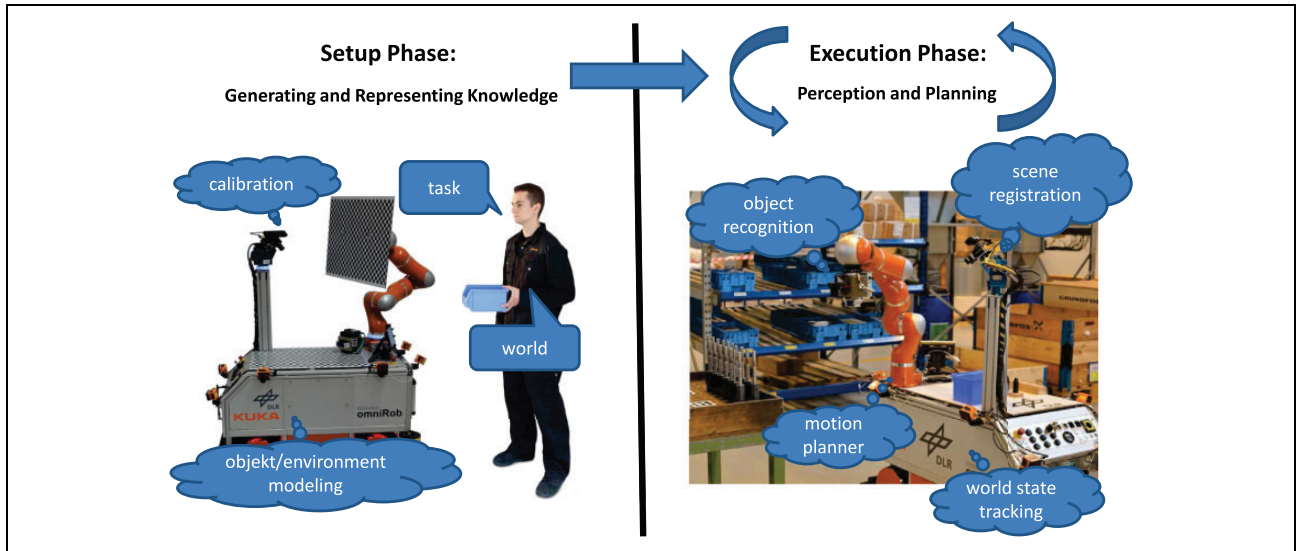


Figure 2. Modules of an autonomous mobile manipulator in the two phases of industrial tasks: setup (left) and execution phase (right).



Figure 3. The DLR omniRob, an autonomous mobile manipulator for fetch and carry operations in industrial environments (top). Further systems applying the proposed concept (bottom).

was evaluated in laboratory setup and in real-world application. Finally, conclusions and an outlook on future work are given.

Related work

Autonomous mobile manipulation has been an actively researched topic for many years. Recently, interest in the field has increased and several commercial systems such as the Willow Garage PR2,¹ Robotnik RB-1 (<http://www.robotnik.eu/manipulators/rb-one/>, 2015), PAL Robotics Tiago (<http://www.tiago.pal-robotics.com/>, 2015), KUKA omniRob III, a predecessor of the KUKA KMR iiwa (http://www.kuka-robotics.com/en/products/mobility/KMR_iiwa/, 2015), Fraunhofer Care-O-bot,³ and rob@work⁴ have become available mostly for researchers but also for industrial customers.

Nevertheless, in literature, only very few publications can be found describing the system concepts and application in the industrial domain. Furthermore, only few researchers have carried out experiments with mobile manipulators in real industrial environments.⁵ Most of the well-published autonomous mobile manipulation systems are from the service robotics domain. However, both domains have to overcome similar challenges. Here, a brief overview of the current state of the art in autonomous mobile manipulation (both, industrial, as well as service robotics) will be presented. A thorough review of past research in the field of autonomous industrial mobile manipulation can be found in Bøgh et al.⁶ and Hvilshøj et al.⁷

In the service robot domain, the PR2 is applied to perform manipulation tasks in a real home by Ciocarlie et al.⁸ The authors state that mobile robots are not able to achieve the level or reliability needed in real living environments. Thus, teleoperating the robot by a human is a vital component of the system concept. In Beetz et al.,⁹ the PR2 fetches and carries pancake mix, plates, and cutlery from drawers and cupboards based on perception for making and serving pancakes.

The rob@work is a prototype of an autonomous mobile manipulator to support human workers in an industrial environment. It consists of a mobile omnidirectional base and can optionally be equipped with various commercially available lightweight arm systems.⁴ Although not many recent publications concerning rob@work are available, its technology (both hardware and software) is based on the Care-o-bot³ service robot family. The Care-o-bot robots are capable of environment modeling, indoor navigation, and executing fetch and carry tasks in a service robotics setting. However, the rob@work system has not carried out industrial tasks in a real factory setting autonomously.

Furthermore, a concept for an autonomous industrial manipulator called “Little Helper” was presented in Hvilshøj et al.,¹⁰ and later improved in Hvilshøj and Bøgh¹¹ and Madsen et al.⁵ The robot is mainly composed of commercial off-the-shelf components which are accessible over a graphical user interface (GUI) for programming the robot. In contrast to our approach, the robot is programmed in the setup phase using a skill-based approach still requiring expert knowledge over the system. The “Little Helper 3” robot took part in two cooperative demonstrations with the system presented in this work. At a demonstration in a Grundfos factory, described by Bøgh et al.,¹² the two systems successfully demonstrated the collaborative production of a component of a pump. In the experiment, the “Little Helper” assembled the rotor pumps based on known object locations.

Another autonomous mobile manipulator focused on service robotics is HERB 2.0 by Srinivasa et al.¹³ The authors propose a modular software concept which is organized by a behavior engine. In contrast to our hierarchical approach, the behavior is encapsulated into three distinct layers. The robot was applied to perform different tasks such as manipulation of a bottle and object recognition of scenes. Caused by the different application domain, there is no distinction between setup and execution phase.

A very similar mobile manipulator was introduced in Zhou et al.,¹⁴ using the same platform, the KUKA omniRob, in an industrial environment, the aerospace industry. This mobile manipulator is programmed employing a GUI allowing the user to select high-level tasks, like sealing and inspection tasks. To assure safety, various sensors for workspace surveillance are integrated in a safety controller. Nevertheless, the contribution of that work is complementary to ours since they focus on human–robot interaction and safety aspects.

System architecture

One of the conclusions from the Amazon Picking Challenge was that system integration and development remain fundamental challenges in robotics.¹⁵ Thus, the system architecture of the robotic platform is described in detail in this section. The first subsection describes the general requirements of a mobile manipulator and our

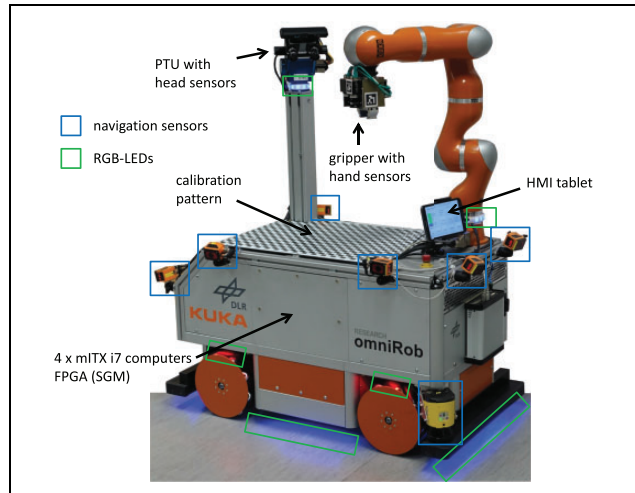


Figure 4. DLR omniRob: KUKA omniRob extended with various sensors, IT components, and human–robot interfaces.

choice of hardware components. The second subsection shows our approach for the software architecture of the mobile manipulator.

Hardware architecture

The robotic system (Figure 4) developed for our approach toward an autonomous mobile manipulator in industrial environments can be subdivided into different classes. These are actuators, consisting of the platform and the arm, sensors, computers, and interfaces to the human worker.

Mobile platform with manipulator. In order to carry out manipulation at different places without the need of a fence around the robot and also in proximity of humans, a mobile platform is required that allows for safe navigation. Further, a torque-controlled manipulator is needed for peg-in-hole operations and for overcoming uncertainties in the perception. The mobile platform, utilized for our work, is a KUKA omniRob. However, the concept is applicable to other platforms as shown in the application section. On top of the KUKA omniRob, a KUKA Lightweight Robot (LWR) 4+ is mounted. This arm is a 7 Degrees of Freedom (DoF) torque-controlled robot with 7 kg payload at full speed. For manipulation, a two-finger parallel gripper from Schunk (PG70) was mounted on the flange of the LWR. The platform has a height of 0.65 m and its surface can be used as workspace or cargo area for transportation tasks. The width and length of the robot are about the same as for Euro-pallets. Therefore, the dimensions of the robot allow for access to most industrial production sites. The Mecanum wheels of the robot are enabling high-precision omnidirectional movements. Power supply is realized by lead batteries allowing for up to 8 h of driving without the need of charging.

Sensor concept. A robotic system needs to perceive its environment in order to fulfill tasks autonomously. Therefore,

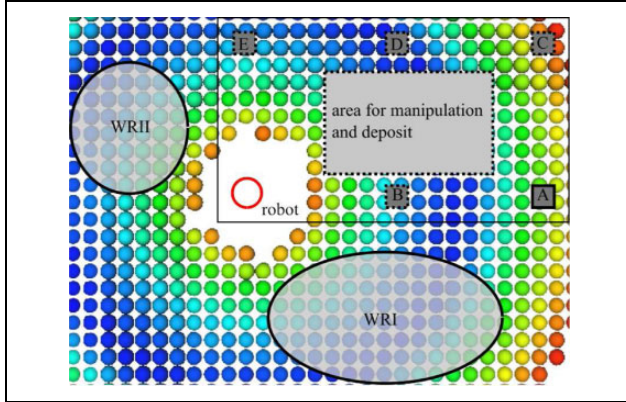


Figure 5. Top view of the platform with reachability evaluation. The platform sizes $0.7 \text{ m} \times 1.2 \text{ m}$. The diameter of a sphere is 0.05 m . Blue spheres indicate best reachability of the robot in terms of the number of possible orientations. Red spheres indicate poor reachability. Resulting suitable WVR I/II are highlighted. WR: working regions.

the sensor concept is designed based on the demands arising from the perceptive abilities of the system. In order to meet these requirements, three sensor subsystems are identified. The main sensors are integrated into the sensor head of the system (see Figure 4). Additional sensors are placed on the hand of the robot. For navigation, a set of sensors are positioned around the platform.

Head sensors. The main sensors of the system are placed on a pan-tilt unit (PTU) which is mounted on a separate pole. This setup is similar to the biological solution of putting the sensors into a head and has several benefits. Due to the independence of the perception from the manipulators, a flexible, task depending, and closing of the action–perception loop is possible. For high-dynamic or high-precision tasks, a short loop closure is possible by observing the manipulation. In static environment or for simple tasks, the perception can prepare for the next step while manipulation is executed on previously sensed information. For manipulation tasks, the coverage of the workspace by the sensor system is fundamental. Putting the sensors on a pole and combining it with a PTU leads to the best possible coverage of the robot’s workspace from a single pose. In order to optimize the position of the sensor head on the mobile platform, workspace analysis for various common tasks was carried out. A reachability map of the robot was created using the capability map workspace representation of Zacharias.¹⁶ The map (see Figures 5 and 6) is the result of sampling the robot’s working range in terms of possible Tool Center Point orientations. Blue areas represent a favorable working region (WR) or area for manipulation and deposit. Figure 5 shows the platform, the robot base, and the reachability map at the height of a standardized work bench from a top view. The position of the favorable WRs and manipulation areas are marked. Further, the five evaluated positions for the PTU pole are marked (A to E).

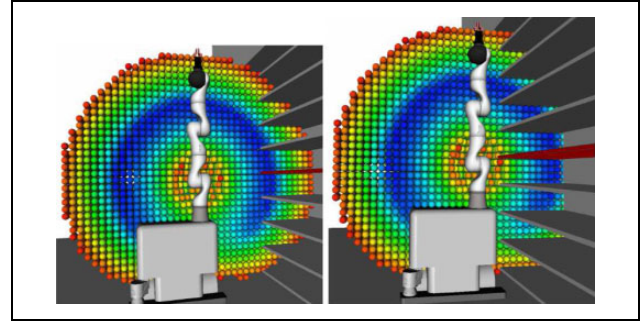


Figure 6. Side view of the platform with reachability evaluation in a shelf scenario. The red shelf is at a height of 1 m .

The sensor characteristics are important for the pose of the sensor system. Taking into account all these factors, we found the most suitable position of the pole at A with a height above ground of 1.4 m . The sensor head is flexible in its sensor equipment due to the use of a standard optical bench. Our standard sensor setup is a high-resolution (1620×1220) stereo camera system which is supported by a structured light projector to add texture on surfaces without natural features. In the configuration with a baseline of 8 cm , this setup has a horizontal field of view of 60° . In ideal conditions (highly and non-repetitive textured surface), the expected depth error in a distance of 1 m is 5 mm . The head sensors are used for modeling the environment, object recognition, and referencing the mobile robot to workstations.

Hand sensors. Since the head sensors cannot closely inspect objects, the minimum size of perceivable object is limited. Therefore, an additional, identical, stereo camera pair is mounted on the gripper of the robot arm. The eye-in-hand cameras allow for additional perspectives, and due to the static transform to the gripper calibration, errors are less critical. To employ the sensors for object in hand localization, wide aperture angles were chosen.

Navigation sensors. Due to the holonomic platform, there is no dedicated motion direction in which sensors for navigation should point. Instead the sensors have to cover the complete proximity of the robot. This design goal is fulfilled by two SICK S300 laser rangefinders mounted on opposing corners of the vehicle, realizing a full 360° view around. However, the laser rangefinders, which are part of the basic KUKA omniRob, only obtain a line scan at a certain height above the floor. Thus, obstacles such as fences or tables cannot be detected. Consequently, additional 3-D sensors, in this case time-of-flight (ToF) cameras, are required that are mounted all around the platform.

IT hardware. The IT hardware has to provide sufficient network bandwidth and computational resources to process the available data of all sensors. Furthermore, to avoid problems with unreliable wireless network, all software

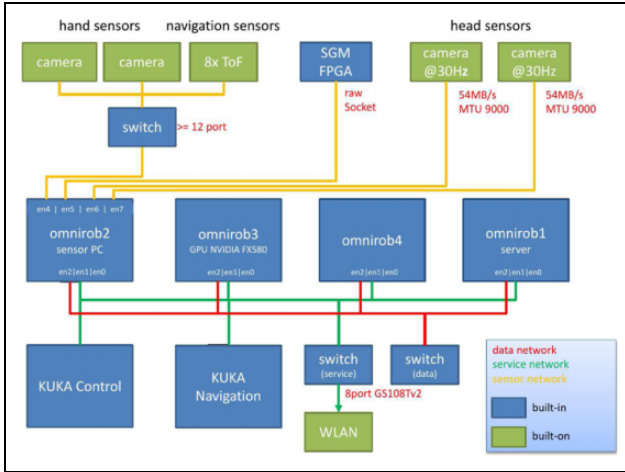


Figure 7. Network structure of the mobile manipulator.

modules have to be executed on the system. Thus, in addition to the robot controller and the navigation computer, which come with the basic system from KUKA, the computing capacity of the platform is increased by adding four computers (mITX, i7, Linux) and the necessary network facilities. In Figure 7, the major IT components and the network connection between them are sketched. One of the computers acts as server for keeping the computers on the same software level. Two parallel networks are used: one for the remote access to the server and one for other communication. The second computer is equipped with four additional network ports connected to the sensors of the system. By providing multiple ports for the sensors, a higher data throughput can be achieved. Collecting all sensor and actuator data on one computer simplifies time stamping. The other two computers are used for computationally costly algorithms. One of these boards is equipped with an additional graphic card which is employed for GPU implementations of algorithms. For stereo processing, a dedicated field-programmable gate array (FPGA) board is installed which provides an implementation of the semi-global matching algorithm from Hirschmüller.¹⁷

Human robot interfaces. An autonomous robotic system in an industrial environment has to interact with the human workers at the site. Many interaction procedures are needed during the training of the robot. The objective is an intuitive interface which enables a natural training similar to that of human workers. There is also the possibility that for some tasks, a cooperation between robot and human worker is necessary. Another important issue is that human workers need to perceive what the robot is currently doing in order to predict what the robot will do next. Therefore, we have multiple hardware devices for human robot interfaces. The compliant robot arm allows for physical interaction between humans and robot. Additionally, to the basic robotic system, we mounted a tablet computer close to the arm. On this display, the state of the robot is visualized to

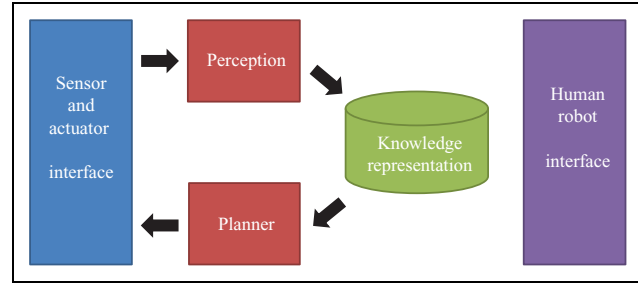


Figure 8. Software module classes of an autonomous mobile manipulator and their relation.

the human worker and buttons for interaction are provided. Furthermore, for simple messages to the humans, like error state, driving direction, and so on, stripes of RGB LEDs were mounted around the robotic platform, the sensor head, and the robot arm (see Figure 4). However, in this work, the interfaces of the robots are used by robotic experts. How to design such interfaces for unskilled workers in an industrial application is beyond the scope of our work.

Software architecture

An autonomous mobile manipulator has to solve different tasks under different conditions involving the robot's hardware and diverse algorithms. Depending on the major impact factors, the environment and the task, different software components have to work together in an appropriate manner. In Figure 8, we depict the major software component classes and their relations.

To meet the flexibility requirements, we propose a fine-grained modular concept. This design enables the robot to solve a wide variety of tasks by reconfiguring the software for the current task.

In the next sections, the major software module categories are introduced, then the communication frameworks between the different modules are described, and finally, the flow control unit orchestrating the different modules in a requirement aware manner is presented.

Module classes. An autonomous robotic system needs a large set of modules to operate in a real industrial environment. These modules can be categorized in module classes (see Figure 8) regarding their functionality.

Interfaces to sensors and actuators. These software modules are the interfaces to the hardware components of the robot. The sensor modules provide the data of the sensors and the functions to configure the sensor. The actuator modules provide access to the actions of the hardware and the information which is provided by the hardware. For instance, the PTU module provides functions to set the pan and the tilt of the unit, and streams the current position of these two DoFs.

Perception. The processing of the sensor data is done in this class of modules. Low-level sensor data processing

modules, such as stereo processing, work directly on the sensor data stream and require only few parameters to work. Usually, they again provide the processed data as stream. Other important modules of the perception class are modelers and object detectors. These more high-level modules deliver different data on a higher abstraction level, for instance, poses of objects or geometric representations of the environment. These modules need a context to deliver reasonable results. Even actions of the whole system can be necessary. These modules usually solve a task and are therefore the central component in their subcomponent of the flow control system.

Knowledge representations. To act autonomously, perception of the environment is not sufficient. For solving tasks, additional information is needed. Therefore, modules are holding, for example, information about the world state, objects, and processes, and are providing them to the system.

Planner. These modules use the knowledge of the robotic system to solve complex tasks. Typical representatives of this module class are global path planners or viewpoint planners for exploration. Usually, the computation effort for these algorithms is very high. Therefore, a common approach is to do preprocessing to speed up the online algorithms.

Human robot interfaces. These modules are used to communicate with humans in the same environment. The basic interface is a status display to show in which state the robot is. For teaching the system, an interactive interface is needed, which can be realized by a GUI. Complex interaction between robot and human worker for working cooperative to solve together a task needs more sophisticated interfaces, including intent detection, for example.

Middleware. To provide a flexible system and keep it usable, we extensively use a modular concept. To establish the connection between these modules, we need a flexible and efficient framework. Due to the different requirements of the modules and especially their interfaces, we decided to use two different middleware. To handle the big data flows from the sensors of the system, we use a shared memory-based approach called SensorNet explained below. For the less data driven and not time critical communication between the higher level modules, we use ROS.¹⁸

SensorNet. All sensors are connected via a middleware denoted SensorNet developed at the DLR. The SensorNet library is designed to provide a small and fast mechanism for distributing streaming data from different sensors such as cameras, pose sensors, and accelerometers. The data are concurrently streamed to different separate applications in real time. SensorNet allows for remote configuration of these devices. The streaming data are provided via a shared memory interface from server to clients (e.g. from the sensor to the applications). Currently, FireWire-, GigE-, and

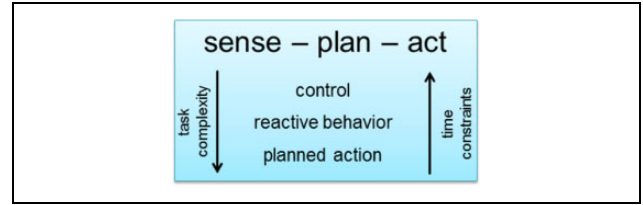


Figure 9. Approaches to close the sense–reason–act loop.

ToF cameras can easily be integrated into SensorNet. The interface to the application provides time-synchronized sensor data calibrated as well as non-calibrated. Furthermore, pose information of the PTU or robot arm can be provided. The client can also configure the server via a configuration channel. The SensorNet is easily configurable and new data channels are easily implemented.

Flow control. To solve a task, the robot has to combine its modules and keep track of the state of the task execution. Depending on the task and the environment, the sense–reason–act loop has to be closed on three different layers which are continuously interconnected. In Figure 9, the correlation between task complexity, speed and different loop closing approaches is depicted.

Some behaviors of the system are implemented on the controller layer. For example, reacting on contact forces has to be solved on that layer to ensure safe interaction with the environment. On this level, only very simple tasks can be solved. The reason part of the loop is realized only by the control algorithm.

More complex but still time critical tasks, for instance, visual servoing or sensor-based collision avoidance, are solved on the reactive layer. In this layer, functionality such as perception or motion generation might be performed in different modules and some cognition has to be done, but each sensor message leads to a resulting action.

Within a very complex task, this direct connection does not exist. The flow control system has to orchestrate up to several thousands of module functionalities. Therefore, a concept similar to a state machine is applied in this layer. Each call of a module functionality can be abstracted as state of the flow. To avoid code duplication and to keep the structure of the state machine manageable, a hierarchical structure is used. This means every state of the state machine can be a state machine itself and can be used in multiple instances in a parent state machine. In addition to the pure state machine, also a data flow between the different states is possible. For some tasks also, parallel execution of states is necessary. For the implementation of this flow control state machine, the SMACH¹⁹ tool available in the ROS frame work is used.

Switching the layer of the sense–reason–act loop approach is realized by encapsulating the closed loops in modules which can be used as states in the more abstract approaches. For example, a position controller is used as

module in the visual servoing module which itself can be used in a state machine to grasp a moving object.

Setup phase: Generating and representing knowledge for industrial manipulation

Nowadays, human experts are needed during the setup phase of a robot. In traditional automation, the robot is programmed to fulfill its task. The programming is increasing in complexity with uncertainties and variations of the task. Programming a robotic system in such a surrounding is very challenging for robotic experts and therefore impossible for an unskilled worker. Thus, as mentioned in the previous chapters, the autonomy of the system should solve this contradiction.

Nevertheless, in the industrial domain, we can benefit from the working routines and the structure of tasks arising. Namely there is, even for human workers, a setup phase in which the worker gets information about how and where to carry out a task. One goal of our approach is to develop a system which allows for training by a standard shop-floor worker. Therefore, multiple modules were developed for acquiring knowledge which is needed during operation of the autonomous mobile manipulator (see the next section) and enables the system to solve its task autonomously. The key point of these modules is that no robotic expert knowledge and no additional hardware is needed to generate the data.

Sensor calibration

In order to combine different measurements in the same coordinate system, knowledge about the positions of the various sensors with respect to each other needs to be obtained. As described in detail in the previous section, the current sensor concept of the mobile manipulator includes two pairs of stereo cameras: one mounted on top of the PTU and one attached to the end effector of the robot arm. Figure 10 presents an overview of the resulting spatial transformations.

In order to be able to calculate depth images, the relative poses between the cameras of each stereo pair, as well as the intrinsic parameters of all involved cameras need to be known. Furthermore, to transform all depth measurements into a common coordinate system, the remaining static transforms marked in green have to be estimated. To this end, a calibration pattern was fixed on the working surface of the mobile manipulator (see Figure 11).

Since the workspace of the PTU is extremely limited, it is not possible to acquire images sufficient for stereo calibration only using this pattern. Therefore, a second calibration pattern can be mounted to the end effector of the robot arm (see Figure 11). Thus, for the cameras mounted on the PTU, images for estimating the intrinsic camera parameters, the relative pose of the cameras and the pan/tilt flange to pan/tilt stereo left transformation are acquired

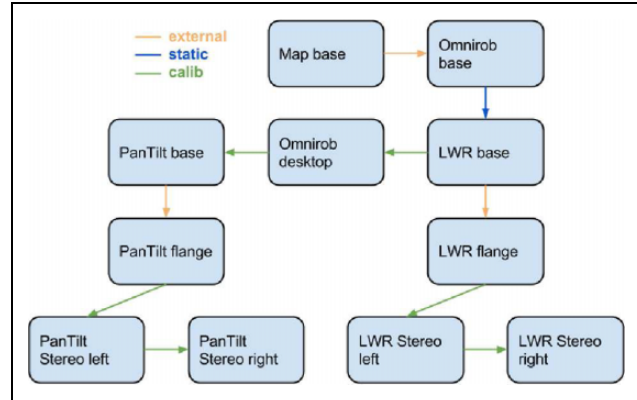


Figure 10. The relevant transformations of the DLR omniRob. Orange: transformations that are retrieved from external measurement systems; blue: static transforms given by CAD data; green: static transformations that need to be estimated during calibration.



Figure 11. The surface and in-hand calibration pattern used for estimating the remaining static transformations and cameras' intrinsic parameters.

by positioning the pattern in predefined poses in front of the pan/tilt stereo system. All remaining images (for the hand mounted stereo cameras and the transformations between PTU and LWR base) are taken from the desktop pattern. Apart from switching between mounting of the cameras and checkerboard on the robot arm, the process of image acquisition is completely automated. The checkerboard detection and parameter estimation itself is done with the DLR CalDe/CalLab toolbox of Strobl et al.²⁰

Object modeling

To carry out manipulation in industrial environments autonomously, a mobile manipulator requires knowledge about the objects to be manipulated. Here, geometric as well as appearance-based object model representations are

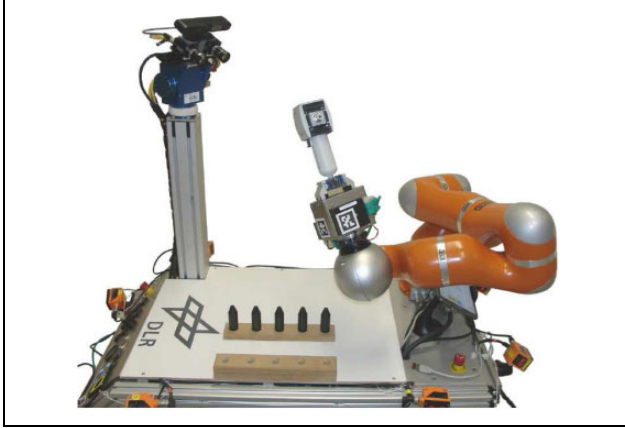


Figure 12. Autonomous object modeling: a pneumatic filter is modeled by grasping the object and perceiving it with the stereo camera on PTU. PTU: pan-tilt unit.

utilized. The different representations are needed as different recognition modules (see the next section) are used depending on the characteristics of the object such as size, shininess, and texture. In this section, we describe how the object models are acquired using the mobile manipulator itself in contrast to modeling the object with a hand-guided scanner system, which requires a human expert and an additional system.

During the robot setup phase, the human worker shows the robot the object of interest which is initially unknown and the robot obtains the required object model fully autonomously without the need of an expert using the approach presented by Kriegel et al.²¹ With the utilized mobile robot, there are two possibilities: either the human places the object onto the robot's working surface and the object is scanned with the stereo cameras on the robot arm or into the gripper and the stereo cameras on the PTU are used (see Figure 12). For the second option, in order to acquire the object models autonomously, the next-best-view (NBV) algorithm as described by Kriegel et al.²¹ is adapted, so that the object is moved instead of the sensor. An NBV represents a sensor viewpoint which provides the best sensory input concerning processing time and model quality for modeling the unknown object (see Scott et al.²² for a good overview). However, this option only works for objects the gripper can actually grasp. For the appearance-based model generation, color images under different lighting conditions are obtained, whereas for the geometric model, depth images under optimal conditions are acquired.

Geometry. In order to autonomously acquire a geometric 3-D object model, NBVs are planned based on the current model in each iteration. Thus, the geometric model generation is tightly coupled with the NBV planning. For instance, once the object is placed onto the robot's working surface or into the gripper and a bounding box is defined, the method of Kriegel et al.²¹ iteratively generates a



Figure 13. A pneumatic filter (left) is modeled when gripped with the mobile robot (middle) and with industrial robot and laser strip (right).

triangle mesh from the range images, registers the scans, plans an NBV, and moves the robot. The process terminates when model coverage and quality required by the 3-D object detection module (see the next section) are reached.

In contrast to autonomous object modeling with an industrial robot and a laser strip,²¹ both the range images and robot poses are significantly noisier resulting in lower quality object models. Additionally, for the second option, some of the objects cannot be firmly grasped with the two-finger gripper, causing the object to drift. Tracking articulated models with a defined kinematic tree as suggested by Schmidt et al.²³ does not work satisfactory as for the last element of the tree, namely the unknown object, no model is given resulting in mismatches and tracking errors. Also, improving local registration by adding color matching as carried out by Krainin et al.²⁴ is not possible for untextured objects as is the case for industrial objects. Furthermore, KinectFusion²⁵ would not perform well as one cannot guarantee that the sensor will always see the object due to robot configuration changes and additionally for the option two where the object is gripped, the only part that changes in the depth image is the object.

Figure 13 shows the model results exemplary for a pneumatic filter. The resulting triangle mesh acquired with the mobile robot proved to be much noisier than with the industrial robot and laser strip system as presented in Kriegel et al.²¹ Note that with the mobile robot, the top of the object cannot be modeled as it is occluded by the gripper. The model errors \bar{e} are 4.68 mm with the mobile robot and 0.77 mm for the industrial robot compared to a ground truth reference model. However, the model quality obtained with the mobile manipulator was still sufficient for object detection.

Appearance. Our appearance-based detection process takes advantage of the fact that many objects (or their parts) show similarities when slightly shifting distance, angle and/or



Figure 14. Rotor caps, which are used in the real application (see the application section), from different viewpoints.

lighting conditions. In practice, such a recognition module is well applicable where background and lighting conditions are changing within reasonable limits, however, viewing angle and distance should not be exceedingly variable. The detection algorithm for rotor caps (depicted in Figure 14) is based upon (but not limited to) grayscale and monocular camera images and provides a 6-D pose-estimation.

Our 2-D object data is represented on the one hand by an image template and on the other hand by a response of patches fed to a Bayes classifier. This twofold approach was adopted because a template matching algorithm is a practical way to cope with statistical inferior observations that nonetheless describe significant parts of the object. These exceptions and anomalies are due to seldom occurring environmental changes like flashes from welders or flickering lights. In those cases, an image of an object may look quite saturated, has odd reflections or simply shows a small irregular area of an object that is very regular everywhere else. Those tweaks have to be applied manually at the moment.

The main idea to collect the data is to take many pictures under different lighting and from varying viewpoints. In the resulting images, we identify keypoints (i.e. AGAST from Mair et al.,²⁶ KLT-corners from Shi and Tomasi,²⁷ centers of Hough transform circles from Kerbyson and Atherton²⁸) and sample patches around them. Since we want to apply a normal Bayes classifier, we need to find appropriate classes and examples accordingly. Thus, we pass the patches through a clustering process in the k-means²⁹ fashion. In an initial state, we treat all patches as single clusters. With a correlation-like measure (in our case the normalized cross-correlation, NCC), we compare all clusters to each other, find the closest similarity with respect to a given threshold and merge the clusters. This results in a set of classes with respect to a threshold as shown in Figure 15. About 10,000 samples are required for our rotor cap object model. The 3-D object model to bind features to patches is preferably provided as a CAD file from the geometric object modeling module (see previous section).

Scene modeling

In order to perform collision-free motion planning with the robot arm (see the next section), a probabilistic voxel space

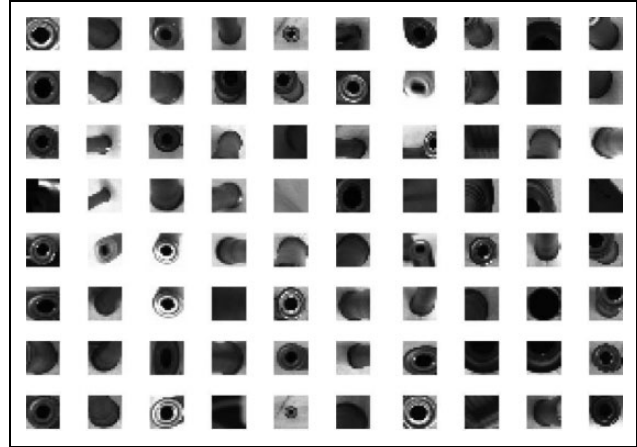


Figure 15. Visualization of clusters: patches of 16×16 pixel were added according to a similarity measure (NCC) and result in depicted examples of clusters. NCC: normalized cross-correlation.

representation is generated for each scene. Obtaining a complete 3-D map of the environment is very costly as a high-resolution model would be required for the complete site, and more importantly, in real production environments, many workstations are movable and might not be exactly in the same position as last time. Thus, we suggest to autonomously create a 3-D model of each scene separately by exploration with the mobile platform as in Kriegel,³⁰ and to register the model to the scene each time interaction is required (see the next section for details on the scene registration).

For scene modeling, we use the stereo camera system on the PTU since it is a lot faster to move the PTU than the robot arm to view in different directions. Before modeling the scene, a human worker needs to attach fiducial markers onto the scene which can later be used for scene registration, remove all objects not relevant for the scene model and teach a platform position in front of each unknown scene. The position teaching can either be done by manually moving the platform to a position in front of the scene or by marking the position on a 2-D map. The position should be selected, so that the robot is approximately centered in front of the scene along the long side with the PTU facing the scene. In contrast to object modeling (see above), due to the use of the PTU without the robot arm, the viewpoint space is very restricted. Thus, NBV candidate generation as suggested for object modeling is not reasonable. For scene modeling, we define the viewpoint space by pan and tilt angles of the PTU and platform movements. Furthermore, a triangle mesh is not a suitable representation as it does not consider sensor uncertainties, and not all parts of the workstation can be modeled due to the limited sensor workspace and could lead to collisions with the robot arm. Therefore, we utilized a probabilistic voxel space (PVS) as described by Kriegel.³⁰ The utilized PVS is needed to identify safe-for-motion areas into which the robot arm can safely move.



Figure 16. For two exemplary scenes (left), a conveyor belt (top) and a shelf (bottom), 3-D scene models are created. The final scene models (right) show that the modeling is able to cope even with the very shiny shelf or conveyor belt. Note the fiducial markers that are attached to the scenes.

During the autonomous scene modeling, the platform and PTU are moved to the selected NBV and a range image is obtained with the stereo system on the PTU. Additionally, for each fiducial marker which is visible in an acquired range image, its position and orientation are saved if the incidence angle is less than 60° . For too high incidence angles, the marker detection does not perform well.³¹ This procedure is repeated until the number of voxels which are free in the space does not change significantly anymore. As the scene is viewed from several positions, the fiducial markers are also viewed multiple times. Therefore, after the NBV algorithm aborts, the positions and orientations of each detected marker are optimized by averaging over all measurements with same marker type. Figure 16 shows two workstations (left), a conveyor belt and a shelf, and the final scene models which were obtained (right). These are two examples of several workstations the mobile robot needed to interact with in a real industrial environment and at a public demonstration (see the application section). The time for autonomous modeling of the workstations, which was performed in a preprocessing step, was between 5 min and 20 min depending on the size of the workstations.

World model

To solve tasks autonomously, the robotic system needs a representation of the environment and itself to keep track of the current world state. The representation of our approach

does not provide a detailed geometric description of the environment, but on a topological level describes the relation between different kinds of world items storing the relevant state of the world. The world model is based on a tree structure, which means that each item has to have a parent item. This structure leads to a natural representation of how manipulations are affecting the world. For instance, if an object is moved, all children of this object are automatically moved with respect to the world frame due to the changed parent frame. There are different types of items in this world representation, which are as follows:

- The most obvious item type is the physical object. This is a representation of an object, holding information such as geometric shape, weight, texture, and orientation constraints to avoid losing parts which are stored in a container.
- Grasps are items which are usually used as children of physical objects storing the grasp-specific parameters such as approach frame, grasp force, and grasp width.
- Bounding box items define a box in the environment. That item type can be used for different purposes, for example, for defining a search volume for an object detection module.
- Obstacle items are regions in the environment where obstacles occur. In industrial applications, some elements of the environment are static and some are dynamic. For instance, the work desk is static, but on its surface, different objects appear. The obstacle items allow for defining such regions which require scene modeling for safe-for-motion path planning.
- Transformation items only hold a transformation. Usually, this item type is used for storing locations or viewpoints in the world representation.
- Semantic items are used to represent item relations. For instance, if an object consists of different parts such as a structure built from different metal profiles, in the assembly process, each profile is added as child to the semantic item assembled part.
- Robot items provide the representation of the robot in the world module. In our approach, two robot items are defined, the robot base and the robot flange.

The world representation is used by various modules which depend on the information of the world state. A computational expensive operation is the generation of the geometric model of the environment from the world representation needed for the geometric planners. To reduce the cost of this step and get a scalable approach, the concept of scenes was introduced. The world representation holds information of the world that the robot needs to know. However, for the geometric planner (see the next section), only the local information is important. Hence, a subset of the world tree is defined as scene in this case, and only this

branch of the tree is passed to the geometric planner. To switch between different scenes, the navigation module of the robot is used. Thus, every branch that is used as scene contains a transformation item which corresponds to a location in the navigation map of the robot. It is used as entry point for the scene switching. After navigating to the location in the navigation map, the robot item including children is set to this entry transformation of the scene, and the geometric planners can load the geometric information of the scene branch. All modules working with geometric data utilize this local scene representation.

During the setup phase, the structure of the world module has to be defined. Therefore, the trainer is guided to collect the necessary data without the need to edit the world model manually. For example, the human worker can add a workstation. This is done by adding a new scene in the world model, teaching a navigation location, and modeling the static environment as described above. If the robot should manipulate an object, all relevant data are collected utilizing the modules described in the previous sections.

Task control

To solve a task such as a pick and place operation, more than one hundred functions from various modules have to be called. Most of these functions are not related to the task instance, but to the general task structure and the robot. As described in the system architecture section, our approach is to organize the complex flow control in a hierarchical manner. The robot-specific problems can be solved on the lower levels of the flow control by a robotic expert. This knowledge is encapsulated into state machines which need few additional information to solve a task. During the setup phase, these high-level state machines are used. This ensures that only the information about the task has to be trained, which is comparable to the training of a human worker. All robot-specific issues are handled by the robotic system itself. For pick and place operations, the following high-level state machines are used:

- goToWorkstation(workstation),
- pickObjectFrom(object_to_pick, object_to_pick_from), and
- placeObjectOn(object_to_place, object_to_place_on).

By sequencing these high-level modules, an unskilled worker could program new tasks for the robot within the fetch and carry domain. It is obvious that a very intuitive interface for such simple task parameters can be found. Of course more complex tasks would need more sophisticated interfaces to program tasks which are beyond the scope of our work. Nevertheless, at the moment, we focus on the fetch and carry task domain, which is one of the most important domains in real industrial application.

Execution phase: Perception and planning for industrial manipulation

Based on the information gained in the previous chapter, the mobile manipulator is able to perform its task fully autonomous during execution phase. Various modules enable the system to be robust with respect to uncertainties and variations in the task and environment. In this chapter, the main components which are needed for the execution of the industrial task are described. First, the modules we employ to detect the task-relevant objects are introduced. Second, the use of the generated knowledge of the environment by local registration to the scene is presented which allows for motion planning to pick or place objects. Finally, keeping the world model of the robot updated ensures correct parametrization of all utilized modules.

Object recognition

A common problem for almost all autonomous manipulating robotic systems is to identify the objects to interact within the vast amount of sensor data from the cameras. Some of the objects of interest in an industrial scenario can be very challenging for camera-based object recognition. Common properties complicating recognition are a lack of texture, shiny surfaces, and insufficient object size. Since there is no single best algorithm, two methods were implemented that cover a broad range of possible object types. The first one is a depth image-based recognition algorithm, that has the advantage of working with any given 3-D Mesh or CAD model of an object without training.³² However, it cannot handle cases in which the possible object cannot be segmented in 3-D data, or when the acquired depth data is too noisy (as in the case of untextured objects or specularities). In such cases, an intensity image-based algorithm proved superior.

3-D object detection and pose estimation. In an industrial environment, it can be assumed that, for a certain task, only a limited number of different objects are relevant. Furthermore, certain knowledge about the environment is presumed in order to enable segmentation of the point cloud data. This prior knowledge can be obtained from plant layouts, CAD models of workstations or be generated by autonomous scene modeling (see previous section).

The developed geometry-based object recognition method is based on the work of Drost et al.³³ A global model for each object is built using a feature similar to the surflet pair features.³⁴ The features are calculated from multiple combinations of model surface point pairs and their corresponding normals. The extracted features are then used as keys in per object hash tables to quickly find similar point pairs with respect to a given candidate pair. Acquired depth images are segmented using the prior knowledge of the environment. Then for each data, cluster candidate point pairs are sampled randomly and similar

model point pairs are retrieved from the hash tables. For each combination of candidate and model point pairs, a rigid transformation can be calculated when considering the corresponding normals. Since the hash table key is not very descriptive, a multitude of possibly conflicting hypotheses will be generated. Therefore, another processing step is necessary, in which hypotheses are clustered and only ones that are supported by a sufficient amount of candidate/model pairs are further processed.

In a last step, quality values for the remaining hypotheses are calculated by rendering the objects in their corresponding poses and pixel wise comparing the resulting depth buffer with the acquired depth data, similar as in Zabulis et al.³⁵ Finally, if the quality of the highest rated hypothesis surpasses a threshold, it is considered to sufficiently explain the data cluster.

Monocular object detection and pose estimation. Sensors and camera setups have different assets. Both 3-D and monocular approaches have comparable qualities regarding the localization of contours, corners, and edges at the *XY*-plane, but stereo algorithms have a tendency to fail at monochromatic, uniform, and especially reflecting surfaces. In general, object recognition and pose estimation is a highly computationally intensive task. Particularly, fitting a model to a point cloud can be challenging in cluttered scenes. When it comes to detecting reflecting metal parts, it is advantageous to prepend a reliable 2-D-preparation step in order to find a region of interest.

The general approach consists of two major phases. As a first step, camera images are analyzed in order to detect the object. This is performed by sampling patches around key-points, comparing those patches with a predefined set of those using suitable classifiers or lookup procedures in data structures. The result is a region of interest with a high probability of an object in it.

As the major next step, the resulting region of interest is inspected for characteristic points (“features”) to determine the objects pose. In the typical case, at least three features have to be visible; however, some object properties including symmetry and a regular shape may reduce the requirements. The methods used are vision-based GPS (VGPS)³⁶ and an implementation of the approach from Gao et al.³⁷ The pose estimation for more complex objects than rotor caps demands supplementary random sample consensus (RANSAC)³⁸ and verification (minimizing reprojection error) steps for robustness. Concerning the rotor caps, an edge-based fitting of an ellipse shape already provided a dependable estimation of the object’s center. The result is a reliable detection and pose-estimation for such a group of objects (see Figure 17 for an example).

Scene registration

The mobile platform autonomously navigates within its environment based on a previously recorded 2-D map

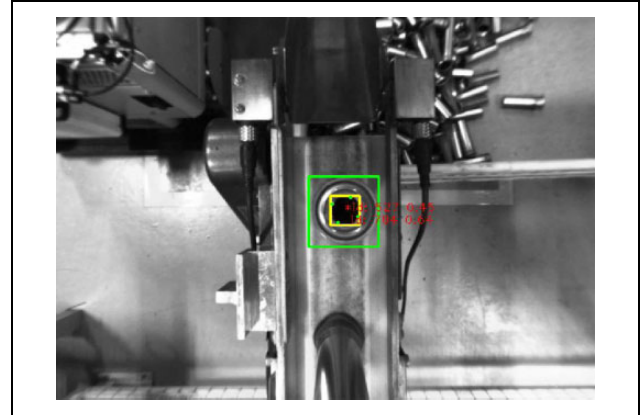


Figure 17. For a rotor cap, viewed from above while on a conveyor belt, the region of interest (green) and subsequently the object’s center (yellow) are detected.

utilizing the laser rangefinders. The navigation approach has been described and evaluated by Röwekamper et al.³⁹ For manipulation, a 2-D projection of the environment is not sufficient. Therefore, each time the robotic system executes a manipulation tasks, the mobile platform registers itself to the 3-D scene models, which are autonomously acquired in the setup phase (see previous section). In order to estimate the pose of the platform with respect to the scene, we suggest the use of fiducial markers which are placed on the workstation. In this case, we use AprilTags presented by Olson.³¹ AprilTags are similar to quick response codes but are designed to encode smaller data allowing for precise detection of its 3-D position with respect to the camera. During the autonomous scene modeling, all observed markers and their pose in the workstation’s coordinate frame are added to the model as described in the previous section. When docking to a scene, the PTU camera is aimed in the direction of the expected markers. If necessary, multiple views of the target area are acquired. Each observed marker is assigned a weight, taking into account the estimated distance to the camera and the angle between camera and marker plane. In order to calculate the rigid transformation between the model and the actual scene, each observed marker is represented by three 3-D points on the corners of its surface and each point is associated with the corresponding marker’s weight. Finally, a weighted minimum least squares solution can be calculated using the method presented by Challis.⁴⁰

Geometric planner

Here, the geometric planner is described which can be utilized for collision-free motion planning with the robot arm (see Figure 18) based on the scene registration and the autonomously acquired scene model (see previous section).

A basic capability of an autonomous robotic system is a module for planning motions. These motions have to meet several hard requirements such as collision-free planning

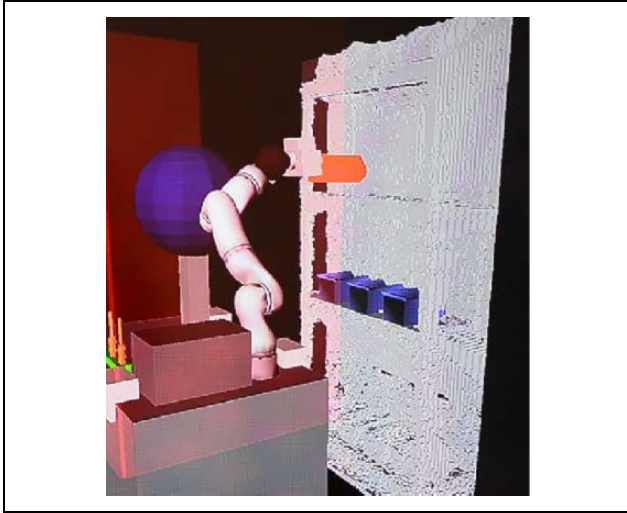


Figure 18. After the platform registers itself with respect to a shelf (see Figure 16 bottom), collision-free motions during manipulation with the robot arm are planned utilizing the safe-for-motion scene model.

and holding of end effector constraints. Further, important features of the motion planner module are planning duration's path quality and execution time. Here, we have a robotic system with 10 DoF. For motion planning in such high-dimensional configuration spaces, sampling-based path planners have been successfully applied.⁴¹ Due to the fact that motions of the platform are much more expensive and less accurate than the arm movements, 10 DoF motions are not used for pick and place operations. If an object is out of reach of the arm, a pure platform movement is performed based on the reachability map of the robot. Thus, the planning problem can be reduced to the 7 DoF configuration space of the robot arm which still demands sampling-based path planning. To pick an object, we can acquire the desired goal transformation of the gripper with respect to the arm by utilizing the world model. By employing an inverse kinematic solver, a goal configuration can be selected. The redundant kinematic of the 7 DoF robot arm can be used to select a configuration which maximizes the distance to joint limits while avoiding obstacles. Every motion of the platform leads to a different occupation of the configuration space and a grasped object leads to a different robot shape depending on the grasp which was selected. Thus, we use single query methods based on rapidly exploring random trees as in Lavalle et al.,⁴¹ namely an RRT-Connect variation by Kuffner and LaValle,⁴² to solve motion planning problems.

World state tracking

The world model introduced in the previous section stores the state of the world. During a pick and place operation, this state is changed by robot actions or perceptions. Therefore, each operation which changes the world's state has to be tracked by the world model in order to keep the model

coherent with the real world. The following operations and their effects on the world state are handled during execution of pick and place tasks:

- **Object recognition:** When an object is localized in the scene, there are different possible effects on the world state. If the detected object is already tracked in the world model, only the object's pose has to be updated. If no corresponding object exists in the world model, a new object has to be added. In this case, a decision has to be made about which parent object should be selected. Usually, a bounding box is used to limit the search space for the object detection module. The parent of this box is a good choice for the object's parent. If no such bounding box exists and the parent cannot be selected task-specific, the scene root has to be selected as object's parent.
- **Scene registration:** The registration to the environment measures the transformation between the robot and the scene root (see above). In the world model, the robot is added as child object to the reference scene, and the measured transformation to the scene is stored.
- **Pick up object:** When an object is grasped, the parent of the object has to be changed to the robot flange in the world model. The transformation to the robot flange is given by the applied grasp.
- **Place object:** When an object is placed onto an object in the scene, the parent of the object changes from the robot flange to the object on which the item was placed.
- **Scene modeling:** When an obstacle item (scene) is explored, the transformation and the structure of the world model tree are unchanged. However, the geometrical information stored in the item is changed, and therefore an environment update of the geometric planner has to be triggered.

Besides these robot operations, the environment can be changed by other robots, machines, or human workers. Depending on the scene, it is reasonable to reset the scene each time the robot enters, since other not tracked changes usually occur and the nominal case is the most likely one.

Application in industrial scenarios

The presented mobile manipulator has been applied to different industrial scenarios: in the lab, at a real production facility at Grundfos A/S in Bjerringbro, Denmark and at a public fair, the Automatica 2014 exhibition. Furthermore, the system concept described in this work was applied to two other platforms, the KUKA KMR iiwa and the DLR Lightweight Rover Unit (LRU).⁴³ The following sections describe these experiments and applications, their goals, and our lessons learned.

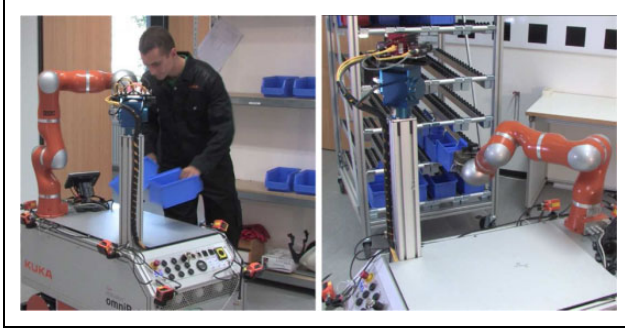


Figure 19. Lab experiments. Left: human worker places SLCs onto robot's working surface. Right: for retrieving empty SLCs from a gravity shelf, the robot platform needs to be moved. SLC: small load carrier.

Lab experiments

In production environments, small objects such as screws and nuts are usually transported in a so-called small load carrier (SLC). By manipulating and transporting these SLCs autonomously, the robot is enabled to assist human workers with various tasks. SLCs are often stored in gravity shelves (see Figure 19, right) from which the worker can take the required parts for assembly. For instance, the utilized gravity shelves are used for the manufacturing process of the KUKA LWR iiwa to allow for sufficient supply of parts at any time. Restocking these shelves with filled SLCs from the warehouse and returning the empty SLCs is currently carried out by workers. For the lab experiment as in Dömel et al.,⁴⁴ which was derived from the LWR iiwa production site at KUKA, at the warehouse a human worker placed full SLCs onto the robot's working surface (see Figure 19, left), the robot transported them to a gravity shelf, placed them into the shelf, retrieved empty SLCs from the shelf, and returned these to a human worker at the warehouse fully autonomously. A full SLC refers to an SLC filled with parts. During the autonomous navigation to the shelf, the robot stopped and detected the position of the SLCs on the robot's working surface, and also created an environment model of the space above the working area to ensure collision-free motion planning. The distance between warehouse and gravity shelf was very low in comparison to the actual production site. At an actual production site, the SLC detection could also be carried out during the navigation for higher efficiency but was not possible due to the smaller lab environment. For retrieving the empty SLCs, which were in the bottom of the shelf, due to the restricted workspace of the robot arm, the platform had to be moved to pull out an SLC (see Figure 19, right) and the arm was switched to low stiffness impedance control mode.

In these first experiments in a controlled lab setting, we were able to show that the modular concept for the system architecture is functional. We also benefit from the hierarchical approach for the flow control system by reusing parts and keeping the task description structured. During

the experiments, we learned some lessons for the further development of the system concept: For grasping the SLCs with the parallel gripper in the restricted space of the gravity shelf, we had to design special fingers. Grasping arbitrary objects with a parallel gripper is not possible. To extend the range of graspable objects, we suggest to utilize a system which allows to exchange the fingers from a set of specialized fingers during operation. In the lab, we could control the lighting conditions of the experiment. Nevertheless, changing lightning conditions were identified as critical point for our perception system, which was stereo-based. Therefore, we decided to integrate different object recognition modules which are better suitable for distinct object classes (shiny, not textured). Through the modularity of our approach, these modules are easily exchangeable to fit the requirement of each task.

Real industrial application

At the pump manufacturer Grundfos A/S in Denmark, the presented mobile manipulator carried out various tasks such as part retrieval from multiple workstations, conveyor handling, part transportation, and delivery of finished parts to the warehouse in order to aid the assembly of a rotor. Figure 20 (left) gives an overview of the 25×15 meter area at Grundfos where the omniRob carried out the logistics tasks.

The omniRob picked up rotor caps from a conveyor belt (spin cell) and a warehouse, and rotor shafts from a warehouse. At the spin cell, the omniRob needed to operate a switch in order to deactivate the conveyor belt (see Figure 20, bottom middle) for picking up parts. If not enough rotor caps were available at the spin cell, the robot picked up missing caps at the warehouse. For transportation, all the parts were stored on a fixture on top of the omniRob. The rotor caps and shafts are delivered to another robot, the Little Helper, which performs the assembly task. For details on the experiment, the parts, and the Little Helper, see Bøgh et al.¹² After picking up the parts, the omniRob collaborated with the Little Helper by delivering them and picking up SLCs which contain the rotors that the Little Helper had already assembled. The full SLCs were then delivered to the rotor warehouse (Station 4) and an empty SLC was passed to the Little Helper. The average cycle time for the omniRob's task was 22 min.

The time schedule for this experiment was a setup phase of 3 days followed by a 1-day execution phase. From the experiments on the real shop floor, we learned that the presented mobile manipulator with the proposed system concept allows for autonomous handling of the parts and carriers in real industrial environments, solving tasks which could not be automated so far.

We also learned some lessons for future improvements of the system. During setup phase, a lot of the time was spent for manual design of the scene models. Despite of the time factor, expert knowledge is needed to create useful

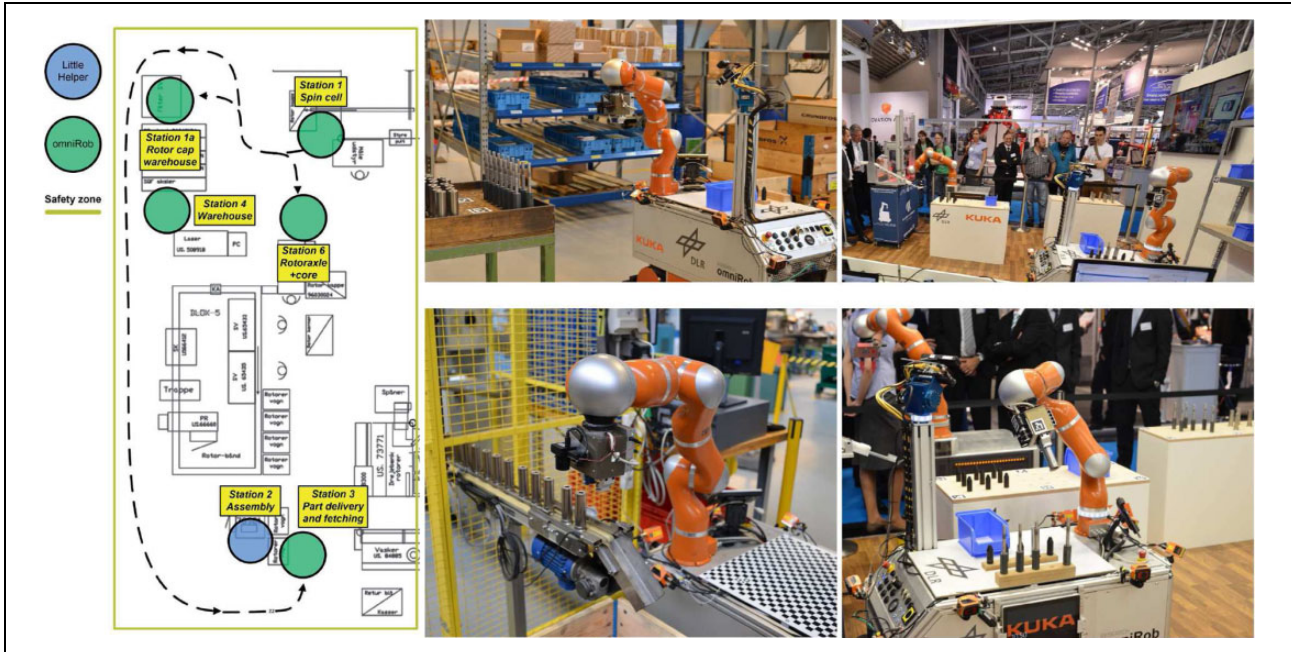


Figure 20. Industrial production site scenario and example stations. Left: map of the production area at Grundfos where the omniRob picks up parts at different workstations, delivers the parts to the Little Helper for assembly, and returns the finished parts to the warehouse, a task which is usually carried out by human workers. The approximate navigation path of the omniRob is indicated by black dashed lines and relevant stations by green circles. Top middle: the robot localizes itself with the stereo camera on the PTU to the *Rotor cap* warehouse station. Bottom middle: in order to pick up parts from a *Spin cell* station, the omniRob needs to operate a switch for deactivating the conveyor belt. Top right: the Grundfos scenario has been scaled to a smaller setup at the Automatica 2014 exhibition. Bottom right: parts are delivered to the assembly station. PTU: pan-tilt unit.

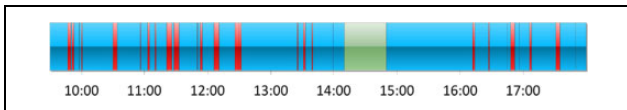


Figure 21. Timeline of the experiment at Grundfos A/S. Blue stands for normal operation, red for downtimes, and green for time spent charging the batteries. Overall, the system spent 87% of the non-charging time in normal operation.

scene models. Therefore, the scene modeling was automated as described in the setup phase section. Even though the omniRob was used only in the logistic domain, the real application showed that the high-level state machines proposed above are not sufficient to solve fetch and carry tasks in real world. In the experiment, the robot has to press a switch before the objects could be picked. Without a robotic expert, another high-level state machine for pressing switches would be necessary. For future work, the set of available skills has to be increased. During the execution phase, errors occurred which could not be handled by the system itself and user interaction was necessary. Specifically, during the 8 h of the experiment, 30 such errors were encountered. See Figure 21 for an overview of their frequency. The three most common error sources were as follows:

- manipulation (10),
- navigation (9), and
- object recognition (2).

For future development, two issues have to be taken into account. First, the execution has to improve its robustness, and second, the system's ability to recover errors has to be improved. At Grundfos, the area where the robot was working was marked and not accessible to the factory workers. For robot-robot or even real human-robot interaction, further safety measures are needed. The mobile manipulator is able to detect humans in the proximity and stop accordingly based on a 2-D plane (from the laser rangefinders) but currently no safety concept exists for humans outside the 2-D plane or for interaction during manipulation with the robot arm. Furthermore, the execution time was significantly longer than it would take a human to perform the same logistics tasks. During the 8-h shift, the robots produced 10 rotor cores. For a human worker, this task only takes several minutes. In contrast to human workers, one could argue that multiple robots could be applied and a robot could work 24 h a day without breaks. Nevertheless, we are still not able to achieve a performance which is comparable to a human worker.

Public demonstration

For the Automatica 2014 exhibition, the scenario from the industrial application at Grundfos A/S in Denmark was mapped to a smaller setup. The part suppliers from the factory were replaced by tables and the press was replaced

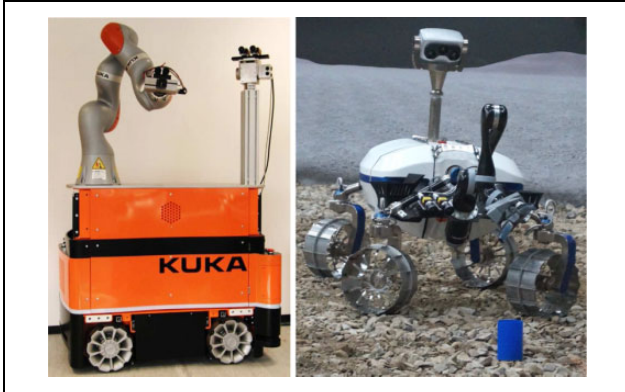


Figure 22. The presented system concept has been transferred to the KUKA KMR iiwa (left) for hosting the Shop Floor Logistics and Manipulation challenge and to the DLR LRU (right) for succeeding at the SpaceBot Camp 2015 for lunar exploration.

by a mock-up (see Figure 20, right top). The tasks for the robots were not changed by these simplifications of the environment. Therefore, the demonstration at the Automatica is comparable with the experiment in the real factory. For the demonstration at the exhibition, the autonomous scene modeling module was used which reduced the setup time to less than a few hours. Furthermore, the system was improved regarding the execution speed and made more reliable. During the exhibition, the task was executed in an endless loop during the opening times. The only error that occurred in this full 4-day demonstration, which caused an intervention by humans, was a broken wire inside the robotic arm. The execution duration could be speed up by a factor of three with respect to the previous experiment, only partially caused by the shorter travel distances.

Transfer to other mobile robots and domains

The hardware and software concept and design presented in this work have been transferred to other mobile robots and also to other domains (see Figure 22). It has been applied to the successor of the KUKA omniRob, namely the KUKA KMR iiwa for carrying out and hosting the Shop Floor Logistics and Manipulation challenge of the European Robotics Challenges (<http://www.euroc-project.eu/>, 2015) (FP7-ICT-608849-EUROC) at the German Aerospace Center (DLR). Five challenger teams from all over Europe are using the mobile manipulator to carry out various tasks ranging from robot–human logistics for aircraft assembly to maintenance operations in hazardous environments and automotive logistics at a car assembly line.

The concept has also been transferred to the DLR LRU for semi-autonomous exploration of moon or mars.⁴³ The LRU succeeded at the SpaceBot Camp 2015 where 10 teams were given the objective to explore an unknown environment with a moon-like planetary surface, locate and fetch two different objects and transport them to a third object for assembly (see Figure 22, right).

The hardware design of the KMR iiwa and the DLR LRU is similar to the mobile manipulator presented in this work. All systems consist of a mobile base to which a force-controlled robot arm with gripper and a PTU are mounted. Additional stereo camera systems are attached to the PTU and the robot arm. Further, the modular software architecture is integrated on computers running on the system. Due to the integration of the same concept on different systems, the administration is reduced and new modules can easily be integrated on different systems.

Conclusion and future work

In this article, the concept of an autonomous mobile manipulator for the industrial domain is presented. In contrast to the service robotics domain, industrial tasks are well defined and the environment is more structured. These characteristics of the domain are exploited by applying a two-phase approach. In the setup phase, knowledge about the task and the environment is generated either autonomously or by the help of a standard shop-floor worker. In the execution phase, an error-prone autonomous execution of the trained task is performed. For both phases, the paradigm of modularization into small functional units is proposed. Combining these modules with a hierarchical flow control system leads to a flexible and still easily usable system. Although we did not completely reach our goals of developing a system comparable to a human worker in the fetch and carry domain, we demonstrated that our system is able to solve problems which have not been automated so far in several experiments and on a real production site. In contrast to traditional robots, the presented mobile manipulator is flexible in its application and is able to perform multiple tasks in a new industrial environment after a short setup phase. To achieve this flexibility and robustness during execution, perception, and planning modules present vital components of the mobile manipulator.

Future work will target the error handling of the system. In the current system design, all uncertainties and errors have to be either solved by the modules or by dedicated fallback strategies. To cope with unforeseeable problems and to detect errors as early as possible, a dedicated instance which detects deviations from the nominal case would be necessary. Ideally, this instance could also recover the systems state to get back to the nominal case. Moreover, an issue raised by the real-world application is the safety of the system while sharing the workspace with human workers. Identifying and more importantly also certifying methods how to collaborate with humans in the workspace in a safe way is one of the key points for bringing autonomous mobile robots to real industrial application. For applications in a real production process, optimizing the execution speed is mandatory.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has partly been supported by the European Commission under contract number FP7-ICT-608849-EUROC and the European Union's H2020 framework programme under grant agreement number 645403 (RobDREAM).

References

- Wyrobek K, Berger E, Van der Loos H, et al. Towards a personal robotics development platform: rationale and design of an intrinsically safe personal robot. In: *IEEE international conference on robotics and automation*, Pasadena, California, May 2008, pp. 2165–2170.
- Borst C, Wimböck T, Schmidt F, et al. Rollin' Justin—mobile platform with variable base. In: *IEEE international conference on robotics and automation*, Kobe, Japan, 11–15 October 2009, pp. 1597–1598.
- Reiser U, Connette C, Fischer J, et al. Care-o-bot 3—creating a product vision for service robot applications by integrating design and technology. In: *IEEE/RSJ IROS*, St. Louis, USA, 11–15 October 2009, pp. 1992–1998.
- Drust M, Dietz T, Pott A, et al. Production assistants: the rob@work family. In: *ISR*, Seoul, Korea, 24–26 October 2013.
- Madsen O, Bøgh S, Schou C, et al. Integration of mobile manipulators in an industrial production. *Industr Robot* 2015; 42(1): 11–18.
- Bøgh S, Hvilshøj M, Kristiansen M, et al. Autonomous industrial mobile manipulation (AIMM): from research to industry. In: *42nd international symposium on robotics*, Munich, Germany, 7–9 June 2011.
- Hvilshøj M, Bøgh S, Skov Nielsen O, et al. Autonomous industrial mobile manipulation (AIMM): past, present and future. *Industr Robot* 2012; 39(2): 120–135.
- Ciocarlie M, Hsiao K, Leeper A, et al. Mobile manipulation through an assistive home robot. In: *IEEE/RSJ IROS*, Vilamoura, Portugal, 7–11 October 2012, pp. 5313–5320.
- Beetz M, Klank U, Maldonado A, et al. Robotic roommates making pancakes—look into perception-manipulation loop. In: *IEEE ICRA workshop on mobile manipulation*, Shanghai, China, 9–13 May 2011, pp. 9–13.
- Hvilshøj M, Bøgh S, Madsen O, et al. The mobile robot little helper: concepts, ideas and working principles. In: *IEEE ETFA*, Mallorca, Spain, 22–25 September 2009.
- Hvilshøj M and Bøgh S. Little Helper—an autonomous industrial mobile manipulator concept. *Int J Adv Robotic Sys* 2011; 8(2). <http://journals.sagepub.com/doi/full/10.5772/10579>
- Bøgh S, Schou C, Ruehr T, et al. Integration and assessment of multiple mobile manipulators in a real-world industrial production facility. In: *ISR/Robotik*, Munich, Germany, 3–6 June 2014, pp. 1–8.
- Srinivasa SS, Berenson D, Cakmak M, et al. Herb 2.0: lessons learned from developing a mobile manipulator for the home. *Proc IEEE* 2012; 100(8): 2410–2428.
- Zhou K, Ebenhofer G, Eitzinger C, et al. Mobile manipulator is coming to aerospace manufacturing industry. In: *IEEE ROSE*, Timisoara, Romania, 16–18 October 2014, pp. 94–99.
- Correll N, Bekris KE, Berenson D, et al. Lessons from the amazon picking challenge. *CoRR abs/1601.05484*. 2016. <http://dblp.uni-trier.de/rec/bibtex/journals/corr/CorrellBBBCHORR16>
- Zacharias F. *Knowledge representations for planning manipulation actions*. PhD thesis, Technische Universität München (TUM), 2011.
- Hirschmüller H. Stereo processing by semiglobal matching and mutual information. *IEEE Trans Pattern Anal Mach Intell* 2008; 30(2): 328–341.
- Quigley M, Conley K, Gerkey B, et al. ROS: an open-source robot operating system. In: *IEEE ICRA workshop on open source software*, Kobe, Japan, 12–17 May 2009.
- Bohren J and Cousins S. The smach high-level executive. *IEEE Robot Autom Magaz* 2010; 17(4): 18–20.
- Strobl KH, Sepp W, Fuchs S, et al. DLR CalDe and DLR CalLab. 2005. http://www.dlr.de/rmc/rm/desktopdefault.aspx/tabid-3925/6084_read-9197
- Kriegel S, Rink C, Bodenmüller T, et al. Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects. *J Real Time Image Proc* 2015; 10(4): 611–631.
- Scott WR, Roth G, and Rivest JF. View planning for automated 3D object reconstruction inspection. *ACM Comput Surv* 2003; 35(1): 64–96.
- Schmidt T, Newcombe R, and Fox D. Dart: dense articulated real-time tracking. In: *RSS*, Rome, Italy, 13–15 July 2014.
- Krainin M, Curless B, and Fox D. Autonomous generation of complete 3D object models using next best view manipulation planning. In: *IEEE international conference on robotics and automation*, Shanghai, China, 9–13 May 2011, pp. 5031–5037.
- Izadi S, Kim D, Hilliges O, et al. Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In: *ACM UIST*, Santa Barbara, USA, 16–19 October 2011, pp. 559–568.
- Mair E, Hager GD, Burschka D, et al. Adaptive and generic corner detection based on the accelerated segment test. In: *ECCV*, Crete, Greece, 5–11 September 2010.
- Shi J and Tomasi C. Good features to track. In: *IEEE CVPR*, Seattle, USA, 21–23 June 1994, pp. 593–600.
- Kerbyson D and Atherton T. Circle detection using hough transform filters. In: *Image processing and its applications*, Edinburgh, UK, 4–6 July 1995, pp. 370–374.
- Lloyd S. Least squares quantization in PCM. *IEEE Trans Inform Theory* 1982; 28(2): 129–137.
- Kriegel S. *Autonomous 3D modeling of unknown objects for active scene exploration*. PhD Thesis, Technische Universität München (TUM), 2015.

31. Olson E. AprilTag: a robust and flexible visual fiducial system. In: *IEEE ICRA*, Shanghai, China, 9–13 May 2011, pp. 3400–3407.
32. Kriegel S, Brucker M, Marton ZC, et al. Combining object modeling and recognition for active scene exploration. In: *IEEE/RSJ IROS*, Tokyo, Japan, 3–7 November 2013, pp. 2384–2391.
33. Drost B, Ulrich M, Navab N, et al. Model globally, match locally: efficient and robust 3D object recognition. In: *IEEE CVPR*, San Francisco, USA, 13–18 June 2010, pp. 998–1005. IEEE.
34. Wahl E, Hillenbrand U, and Hirzinger G. Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification. In: *3DIM*, Banff, Canada, 6–10 October 2003, pp. 474–481.
35. Zabulis X, Lourakis M and Koutlemanis P. 3D object pose refinement in range images. In: *ICVS*, Copenhagen, Denmark, 6–9 July 2015.
36. Burschka D and Hager G. V-GPS (SLAM): vision-based inertial system for mobile robots. In: *IEEE ICRA*, New Orleans, USA, 26–30 April 2004, pp. 409–415.
37. Gao XS, Hou XR, Tang J, et al. Complete solution classification for the perspective-three-point problem. *IEEE Trans Pattern Anal Mach Intell* 2003; 25(8): 930–943.
38. Fischler MA and Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 1981; 24(6): 381–395.
39. Röwekämper J, Sprunk C, Tipaldi GD, et al. On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In: *IEEE/RSJ IROS*, Vilamoura, Portugal, 7–11 October 2012, pp. 3158–3164.
40. Challis JH. A procedure for determining rigid body transformation parameters. *J Biomech* 1995; 28(6): 733–737.
41. Lavelle SM and Kuffner JJ Jr. Rapidly-exploring random trees: progress and prospects. *Algorithmic and computational robotics: new directions* 2000; 293–308. <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.7457>
42. Kuffner JJ and LaValle SM. RRT-connect: an efficient approach to single-query path planning. In: *IEEE ICRA*, 2000, pp. 781–787.
43. Schuster MJ, Brand C, Brunner SG, et al. The LRU rover for autonomous planetary exploration and its success in the SpaceBotCamp challenge. In: *ICARSC*, Braganca, Portugal, 4–6 May 2016.
44. Dömel A, Kriegel S, Brucker M, et al. Autonomous pick and place operations in industrial production. In: *URAI*, Goyang, South Korea, 28–30 October 2015.